Mathematics in Intelligent Systems

ALTERNATING DIRECTION METHOD OF MULTIPLIERS (ADMM)

- **Rohith Ganesan**

TABLE OF CONTENTS

Abstract

The alternating direction method of multipliers (ADMM) is an algorithm that solves convex optimization problems by breaking them into smaller pieces, each of which are then easier to handle. *Hereby reducing the time of solving a problem. It is a popular approach for solving optimization problems that are non-smooth and with hard constrains. It has been applied to various computer graphics applications, including physics simulation, geometry processing and image processing. However, ADMM can take a long time to converge to a solution of high accuracy.* The alternating direction method of multipliers (ADMM) has emerged as a powerful technique for large-scale structured optimization. Despite many recent results on the convergence properties of ADMM, a quantitative characterization of the impact of the algorithm parameters on the convergence times of the method is still lacking. *In this report you will find the background details as well as some examples on distributed optimization with the alternating direction methods of multipliers.*

Introduction

The alternating direction method of multipliers is a powerful algorithm for solving structured convex optimization problems. The generalization of ADMM's usage is in solving convex optimization problems where the data can be arbitrarily large. While the admm method was introduced for optimization in the 1970's, its origins can be traced back to techniques for solving elliptic and parabolic partial difference equations developed in the 1950's . Admm enjoys the strong convergence properties of the method of multipliers and the decomposability property of dual ascent, and is particularly useful for solving optimization problems that are too large to be handled by generic optimization solvers. The method has found a large number of applications in diverse areas such as compressed sensing, regularized estimation , image processing , machine learning , and resource allocation in wireless networks. This broad range of applications has triggered a strong recent interest in developing a better understanding of the theoretical properties of admm. The generalization of admm's usage is in solving convex optimization problems where the data can be arbitrarily large.

Generally, such issues are solved by using parallel versions of algorithms to distribute the work-load across multiple processors, thus speeding up the optimization. But our traditional optimization algorithms are not suitable for parallel computing, so we must use a method that is. Such a method would have to decentralize the optimization; one good way to do this is to use the alternating direction method of multipliers (admm). This convex optimization algorithm is robust and splits the problem into smaller pieces that can be optimized in parallel.

We will first give some background on admm, then describe how it works, how it is used to solve problems in practice, and how it was implemented. Then, we discuss common problems that admm is used to solve and how they were implemented as general solvers in the admm library. Finally, we discuss our results on adaptive step-size selection and draw conclusions on the potential for this to be used in practice. We end by discussing potential future work on this solver library and adaptive step-sizes.

After briefly surveying the theory and history of the algorithm, we discuss applications to a wide variety of statistical and machine learning problems of recent interest, including the lasso, basis pursuit, covariance and many others.

Background

Smooth vs Non Smooth

Smooth functions have a unique defined first derivative (slope or gradient) at every point. Graphically, a smooth function of a single variable can be plotted as a single continuous line with no abrupt bends or breaks. All the examples you've seen so far in this section have been smooth.

Non-smooth functions include non-differentiable and discontinuous functions. functions with first derivatives with undefined regions are called non-differentiable. Graphs of non-differentiable functions may have abrupt bends. The absolute value of a variable, ABS(X), is an example of a non-differentiable expression, as illustrated in the following graph:

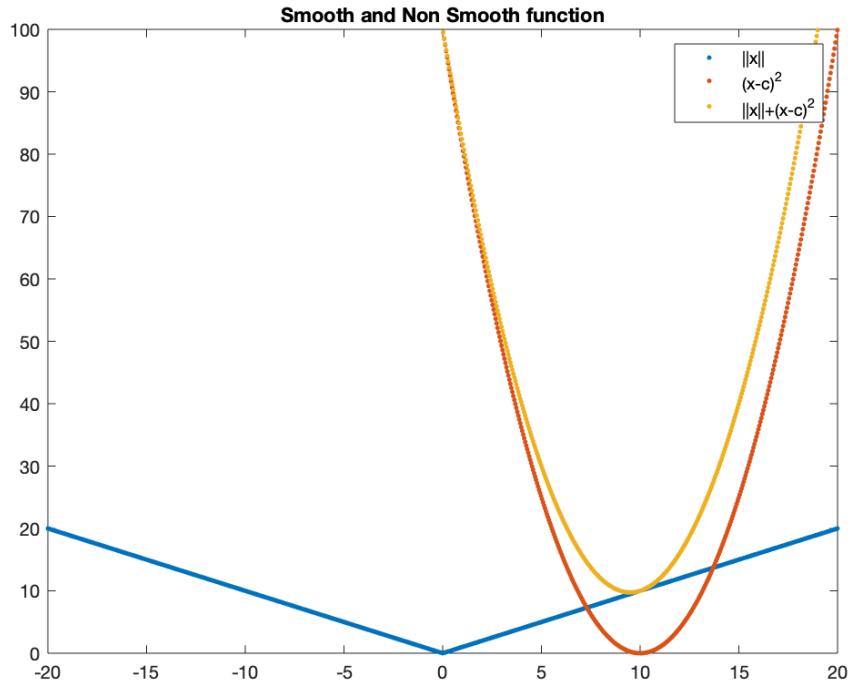


FIGURE 1

The blue line represents the non-smooth function. The red line represents the smooth function. The blue line is obtained by adding the smooth function and non-smooth function. This is how we make a non-smooth function smooth so that it behaves well while finding the optimal point.

Non Smooth function: $f(x) = |x| = abs(x)$

Smooth function: $f(x) = (x - c)^2$

Smooth +Non Smooth function: $f(x) = |x| + (x - c)^2$

Finding the optimal point:

$$x^* = arg\min_x f(x) = \lambda|x| + (x - c)^2$$

$$f'(x) = \lambda sign(x) + 2(x - c) = 0$$

$$\frac{d}{dx}|x| = sign\,(x), \forall x \neq 0$$

$$x^* = c - \frac{\lambda}{2} sign(x^*) = c - \frac{\lambda}{2} sign(c)$$

Solution point is always between 0 and location of vertex of parabola which is c.

If $|c| > \frac{\lambda}{2}$ $x^* = c - \frac{\lambda}{2} sign(c)$

Else

$x^* = 0$

Let's consider an example:

$$f(x) = \lambda|x| + (x - c)^2, \quad \lambda=1, c=\frac{1}{4}$$

Solution is expected between 0 and $c = \frac{1}{4}$

$$But \text{ as per formula, } x^* = c - \frac{\lambda}{2} sign(c) = \frac{1}{4} - \frac{1}{2} \times 1 = -\frac{1}{4}$$

Solution does not have sign of c.

NB: Shrink towards zero but do not cross.

ADMM

Form-1

$$\min_x f(x) + g(x)$$

$x, u, z \in R^n$,

$\lambda \in \mathrm{R}^+$ a hyper parameter

$$x^{k+1} := prox_{\lambda f}\left(z^k - u^k\right)$$
$$z^{k+1} := prox_{\lambda g}\left(x^{k+1} + u^k\right)$$
$$u^{k+1} := u^k + x^{k+1} - z^{k+1}$$

z-surrogate variable to x-variable to be minimised

u-lagrangian multipliers

If f(x) and g(z) are differentiable, Then

$$x^{k+1} = \underbrace{\arg\min}_{x} f(x) + \frac{1}{2\lambda}\left\|x - \left(z^k - u^k\right)\right\|_2^2$$

$$\Rightarrow x^{k+1} \text{ is the solution of } \nabla f(x) + \frac{1}{\lambda}\left(x - \left(z^k - u^k\right)\right) = 0 \text{ vector}$$

$$z^{k+1} = \underbrace{\arg\min}_{z} g(z) + \frac{1}{2\lambda}\left\|z - \left(x^{k+1} + u^k\right)\right\|_2^2$$

$$\Rightarrow z^{k+1} \text{ is the solution of } \nabla f(z) + \frac{1}{\lambda}\left(z - \left(x^{k+1} + u^k\right)\right) = 0 \text{ vector}$$

$$u^{k+1} = u^k + x^{k+1} - z^{k+1}$$

x,z (Hight dimension parabola)

Increase number of variables from one set to three set , solve one set at a time,

$$\min f(x) + g(z)$$
$$subject\ to \quad x - z = 0$$

$$x - z = \mathbf{0} \text{ vector} \triangleq \begin{pmatrix} x_1 - z_1 = 0 \\ x_2 - z_2 = 0 \\ \vdots \\ x_n - z_n = 0 \end{pmatrix}$$ Vector form

$$L_\rho(x, y, z) = f(x) + g(z) + y^T(x - z) + \left(\rho/2\right)\|x - z\|_2^2$$

Here,

- y is lagrangian multiplier term

i.e. $y_1\left(x_1 - z_1\right) + y_2\left(x_2 - z_2\right) + \ldots + y_n\left(x_n - z_n\right)$

- $\left(\rho/2\right)$ is augmented langrangian term

Breaking down the problem:

$$x^{k+1} := \arg\min_x L_\rho\left(x, z^k, y^k\right)$$
$$z^{k+1} := \arg\min_z L_\rho\left(x^{k+1}, z, y^k\right)$$
$$y^{k+1} := y^k + \rho\left(x^{k+1} - z^{k+1}\right)$$

$$L_\rho(x, y, z) = f(x) + g(z) + y^T(x - z) + \frac{\rho}{2}\|x - z\|_2^2$$
$$= f(x) + g(z) + \underbrace{y^T x}_{\langle y, x\rangle} - \underbrace{y^T z}_{\langle y, z\rangle} + \frac{\rho}{2}(x - z)^T(x - z)$$

The lagrangian function is solved 'one set' of variable *at a time.*
In the $k+1$ *th iteration for computing* x, *we assume* y^k, z^k *is known .*
x^k, y^k, z^k *are values of* x, y *and* z *obtained in* k^{th} *iteration*

Consider minimization *of Lagrangian function L w.r.t* x,
assuming $y = y^k$ *and* $z = z^k$ *is known.*

Lagrangian function

$$L_\rho(x) = f(x) + \underbrace{g(z^k)}_{cons\tan t} + \underbrace{x^T y^k}_{\langle y\ ,\ x\rangle} - \underbrace{\left(y^k\right)^T z^k}_{cons\tan t} + \frac{\rho}{2}(x - z^k)^T(x - z^k)$$

Omiting constant terms from objective function, we get

$$L_\rho(x) = f(x) + \underbrace{x^T y^k}_{\langle y\ ,\ x\rangle} + \frac{\rho}{2}(x - z^k)^T(x - z^k)$$

So the update for x can be written as

$$x^{k+1} = \arg\min_x \left(f(x) + \underbrace{\left\langle y^k, x \right\rangle}_{y^T x} + \frac{\rho}{2}\left\| x - z^k \right\|_2^2 \right)$$

You can think that , you obtained x^{k+1} *by solving* $\frac{\partial L_\rho(x)}{\partial x} = 0$ *vector*

We assume x and y vectors are known.

We rewrite Lagrangian as follows:-

$$L_\rho(z) = \underbrace{f(x^{k+1})}_{cons\tan t} + g(z) + \underbrace{\left(y^k\right)^T x^{k+1}}_{Cons\tan t} - \left(y^k\right)^T z + \frac{\rho}{2}(x^{k+1} - z)^T(x^{k+1} - z)$$

Omiting constant terms from objective function, we get

$$L_\rho(z) = g(z) - \underbrace{\left(y^k\right)^T z}_{\langle y\ ,\ z\rangle} + \frac{\rho}{2}(x^{k+1} - z)^T(x^{k+1} - z)$$

$$z^{k+1} = \arg\min_z \left(g(z) - \left\langle y^k, z \right\rangle + \frac{\rho}{2}\left\| x^{k+1} - z \right\|_2^2 \right)$$

You can think that , you obtained z^{k+1} *by solving* $\frac{\partial L_\rho(z)}{\partial z} = 0$ *vector*

y vector elements are lagrangian multipliers.

The iterative algorithm proceeds towards (x^*, z^*, y^*) which is a saddle point of L.

This means , w.r.t x, z variables the lagrangian function L is minimum at (x^*, z^*, y^*).

Also, w.r.t y variables the lagrangian function L is maximum at (x^*, z^*, y^*).

$So,$ y vector is updated towards the maximum gradient of L with respect to y

This direction is given by $\frac{\partial L}{\partial y}$

$$L_\rho(y) = f(x^{k+!}) + g(z^{k+1}) + y^T(x^{k+1} - z^{k+1}) + \frac{\rho}{2}\left\|x^{k+1} - z^{k+1}\right\|_2^2$$

$Omiting$ constant terms from objective function, we get $L_\rho(y) = y^T(x^{k+1} - z^{k+1})$

$$\frac{\partial L_\rho(y)}{\partial y} = (x^{k+1} - z^{k+1}) \equiv gradient \text{ of L w.r.t y}$$

So to move towards maximum , y is moved along gradient direction from the current position

$$\therefore y^{k+1} = y^k + (step\_size) \times \left(gradient \text{ of L w.r.t y}\right)$$

$$y^{k+1} = y^k + \rho\left(x^{k+1} - z^{k+1}\right) \quad . \text{ We asumed } step\_size \text{ as } \rho$$

We combine 2nd and 3rd term in the first two optimization into a single term,

$$x^{k+1} = \arg\min_x \left(f(x) + \left\langle y^k, x \right\rangle + \frac{\rho}{2}\left\|x - z^k\right\|_2^2 \right)$$

$$z^{k+1} = \arg\min_z \left(g(z) - \left\langle y^k, z \right\rangle + \frac{\rho}{2}\left\|x^{k+1} - z\right\|_2^2 \right)$$

$$y^{k+1} := y^k + \rho\left(x^{k+1} - z^{k+1}\right)$$

On transforming

$$x^{k+1} = \arg\min_x \left(f(x) + \frac{\rho}{2}\left\|x - z^k + \frac{1}{\rho}y^k\right\|_2^2 \right)$$

$$z^{k+1} = \arg\min_z \left(g(z) + \frac{\rho}{2}\left\|x^{k+1} - z + \frac{1}{\rho}y^k\right\|_2^2 \right)$$

$$y^{k+1} = y^k + \rho\left(x^{k+1} - z^{k+1}\right)$$

We verify it by expanding it:

Consider first optimization :

$$Let\ L_\rho(x) = f(x) + \frac{\rho}{2}\left\| x - z^k + \frac{1}{\rho} y^k \right\|_2^2$$

$$= f(x) + \frac{\rho}{2}\left(\left(x - z^k\right) + \frac{1}{\rho} y^k \right)^T \left(\left(x - z^k\right) + \frac{1}{\rho} y^k \right)$$

$$= f(x) + \frac{\rho}{2}\left\{ \left(x - z^k\right)^T \left(x - z^k\right) + 2\left(x - z^k\right)^T \frac{1}{\rho} y^k + \left(\frac{1}{\rho} y^k \right)^T \left(\frac{1}{\rho} y^k \right) \right\}$$

$$= f(x) + \frac{\rho}{2}\left(x - z^k\right)^T \left(x - z^k\right) + x^T y^k + \text{constant}$$

On rearranging

$$L_\rho(x) = f(x) + x^T y^k + \frac{\rho}{2}\left\| x - z^k \right\|_2^2$$

$$So,\ x^{k+1} = \arg\min_x f(x) + \left\langle x, y^k \right\rangle + \frac{\rho}{2}\left\| x - z^k \right\|_2^2$$

Consider second optimization term :

$$Let\ L_\rho(z) = g(z) + \frac{\rho}{2}\left\| x^{k+1} - z + \frac{1}{\rho} y^k \right\|_2^2$$

$$= g(z) + \frac{\rho}{2}\left(x^{k+1} - z + \frac{1}{\rho} y^k \right)^T \left(x^{k+1} - z + \frac{1}{\rho} y^k \right)$$

$$= g(z) + \frac{\rho}{2}\left\{ \left(x^{k+1} - z\right)^T \left(x^{k+1} - z\right) + 2\left(x^{k+1} - z\right)^T \frac{1}{\rho} y^k + \left(\frac{1}{\rho} y^k \right)^T \left(\frac{1}{\rho} y^k \right) \right\}$$

$$= g(z) + \frac{\rho}{2}\left(x^{k+1} - z\right)^T \left(x^{k+1} - z\right) - z^T y^k + \text{constant}$$

On rearranging

$$L_\rho(z) = g(z) - z^T y^k + \frac{\rho}{2}\left\| x^{k+1} - z \right\|_2^2$$

$$So, z^{k+1} = \arg\min_z g(z) - \left\langle z, y^k \right\rangle + \frac{\rho}{2}\left\| x^{k+1} - z \right\|_2^2$$

Now simplifing it,

With

$$u^k = \frac{1}{\rho} y^k, \quad \lambda = \frac{1}{\rho}$$

Hence,

$$x^{k+1} = \arg\min_{x}\left(f(x) + \frac{\rho}{2}\left\| x - z^{k} + \frac{1}{\rho} y^{k} \right\|_2^2 \right)$$

$$z^{k+1} = \arg\min_{z}\left(g(z) + \frac{\rho}{2}\left\| x^{k+1} - z + \frac{1}{\rho} y^{k} \right\|_2^2 \right)$$

$$y^{k+1} = y^{k} + \rho\left(x^{k+1} - z^{k+1} \right)$$

Start with augmented Lagrangian function in the following form,

$$L_{\lambda}(x,u,z) = f(x) + g(z) + \left(\frac{1}{2\lambda}\right)\|x - z + u\|_2^2$$

instead of

$$L_{\rho}(x,y,z) = f(x) + g(z) + y^{T}(x - z) + \left(\rho/2\right)\|x - z\|_2^2$$

Form-2

$$\begin{aligned} &\min \quad f(x) \\ &subject \;\; to \;\; x \in C \end{aligned}$$

This is converted into,

$$\begin{aligned} &\min \quad f(x) + g(z) \\ &subject \;\; to \; x - z = 0 \end{aligned}$$

Where, Indicator function is

$$g(z) = \begin{Bmatrix} 0 & if \; z \in C \\ \infty & otherwise \end{Bmatrix}$$

Augmented Lagrangian function can be written as

$$L_{\rho}(x,z,u) = f(x) + g(z) + \left(\frac{1}{2\lambda}\right)\|x - z + u\|_2^2$$

$$x^{k+1} = \arg\min_x \left(f(x) + \frac{1}{2\lambda} \left\| x - z^k + u^k \right\|_2^2 \right)$$

$$z^{k+1} = \Pi_C \left(x^{k+1} + u^k \right)$$ Π_C stands for Projection onto Convex set C

$$u^{k+1} = u^k + x^{k+1} - z^{k+1}$$

Explanation for projection

$$z^{k+1} = \Pi_C \left(x^{k+1} + u^k \right)$$

Consider augmented Lagrangian

$$L_\rho(x, z, u) = f(x) + g(z) + \left(\frac{1}{2\lambda} \right) \left\| x - z + u \right\|_2^2$$

Assume x^{k+1} *and* u^k *is known. Then*

$$L_\rho(x^{k+1}, z, u^k) = f(x^{k+1}) + g(z) + \left(\frac{1}{2\lambda} \right) \left\| x^{k+1} - z + u^k \right\|_2^2$$

Omitting constant terms

$$L_\rho(x^{k+1}, z, u^k) = g(z) + \left(\frac{1}{2\lambda} \right) \left\| x^{k+1} - z + u^k \right\|_2^2$$

But, g(z) is an indicator function. It is minimum when z $\in$ C.

Second term is minimum when z $= x^{k+1} + u^k$.

z $= x^{k+1} + u^k$ may not be $\in$ C. If it is not, g(z) is infinity.

So, we project $x^{k+1} + u^k$ into C

Applications of ADMM

Basis pursuit

BP finds the least l1- norm solution of the underdetermined linear system Ax = b and is used, for example, in compressed sensing for reconstruction.

A good proxy for finding the sparsest solution to an underdetermined system of equations Ax = b is to solve:-

minimize $\|x\|_1$ subject to $Ax = b$

ADMM formulation:

minimize $f(x) + g(z)$ subject to $x - z = \mathbf{0}$ vector

where,

$f(x) = \|x\|_1$

$$g(z) = \begin{cases} 0, & Az = b \\ \infty & \text{otherwise} \end{cases}$$

Suppose we got approximate z =z_{app}in an iteration and $Az_{app} \neq b$

What we need is $A(z_{app} + e) = b$

$$Ae = b - Az_{app} \Rightarrow e = pinv(A) \times \left(b - Az_{app}\right)$$

$$\Rightarrow z^{k+1} = z_{app} + e = z_{app} + pinv(A) \times \left(b - Az_{app}\right)$$

Augmented lagrangian is

$$L(x,z,u) = \|x\|_1 + g(z) + \frac{1}{2\lambda}\|x - z + u\|_2^2$$

$$x^{k+1} = \arg\min_x \|x\|_1 + \frac{1}{2\lambda}\left\|x - z^k + u^k\right\|_2^2$$

$$= S_\lambda\left(z^k - u^k\right)$$

$$\min_x f(x) = \|x\|_1 + \frac{1}{2\lambda}\|x - c\|^2$$

$$sign(x) + \frac{1}{\lambda}(x - c) = 0 \Rightarrow x^* = c - \lambda \times sign(c)$$

$$x^* = S_\lambda(c) = \begin{cases} c - \lambda \times sign(c), & \text{if } \lambda \le |c|, \\ 0, & \text{if } \lambda > |c| \end{cases}$$

$$z^{k+1} = \arg\min_x g(z) + \frac{1}{2\lambda}\left\|x^{k+1} - z + u^k\right\|_2^2$$

$$z^{k+1} = P_{Az=b}\left(x^{k+1} + u^k\right) = \left(x^{k+1} + u^k\right) + pinv(A) \times \left(b - A\left(x^{k+1} + u^k\right)\right)$$

$$u^{k+1} = u^k + \left(x^{k+1} - z^{k+1}\right)$$

http://pwp.gatech.edu/ece-jrom/wp-content/uploads/sites/436/2017/07/16-admm.pdf

Plot:

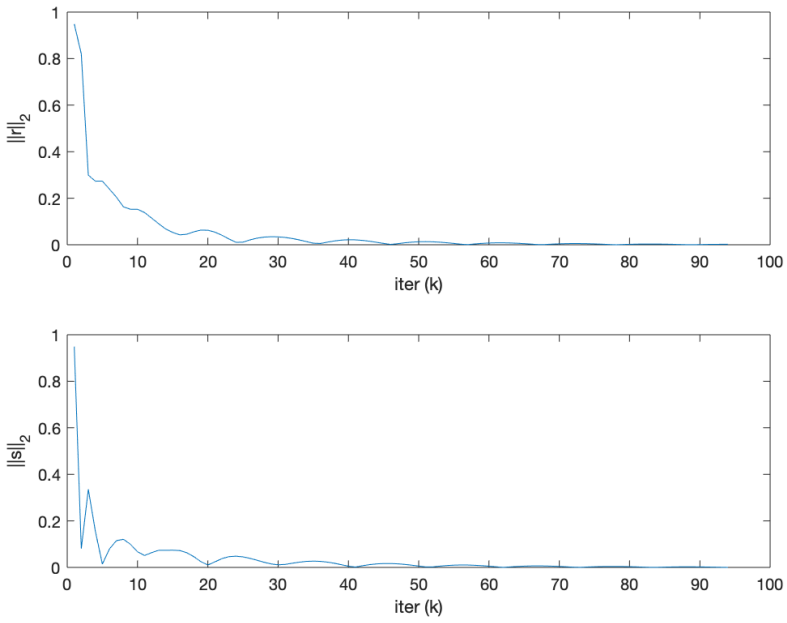


FIGURE 2

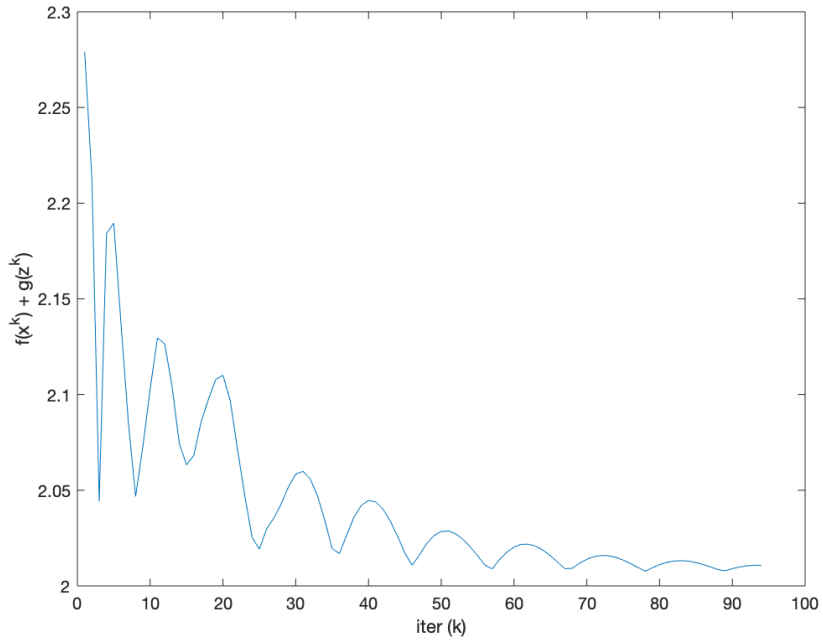


FIGURE 3

Matlab code:

```
clc;
clear all;
close all;

% basis_pursuit  Solve basis pursuit via ADMM
%
% [x, history] = basis_pursuit(A, b, rho, alpha)
%
% Solves the following problem via ADMM:
%
%   minimize     ||x||_1
%   subject to   Ax = b
%
% The solution is returned in the vector x.
% rho is the augmented Lagrangian parameter.
rand('seed', 0);
randn('seed', 0);
n = 10;
m = 6;
A = randn(m,n);
x = sprandn(n, 1, 0.1*n);
b = A*x;
xtrue = x;
%calling basis_pusuit function
[x z u r_norm s_norm g obj] = basis_pursuit(A, b, 1.0, 1.0)
```

```
% for printing  'x^(k+1)','z^(k+1)','u^(k+1)'
fprintf('%20s\t%10s\t%10s\t \n', 'x^(k+1)','z^(k+1)','u^(k+1)');
for i =1:length(x)
fprintf('%20f\t%10.4f\t%10.4f\t \n', x(i),z(i),u(i));
end
figure
%plotting difference between current z and old z
subplot(2,1,1)
plot(1:g,r_norm)
ylabel('||r||_2'); xlabel('iter (k)');
%plotting difference between z and x
subplot(2,1,2)
plot(1:g,s_norm)
ylabel('||s||_2'); xlabel('iter (k)');
figure
%plotting norm with iteration
plot(1:g,obj)
ylabel('f(x^k) + g(z^k)'); xlabel('iter (k)');
function [X_hat z u r_norm s_norm g obj] = basis_pursuit(A, b, rho, alpha)
MAX_ITER = 1000;
[m n] = size(A);
x = zeros(n,1);
z = zeros(n,1);
u = zeros(n,1);
for k = 1:MAX_ITER
    %giving tolerance value
    RELTOL   = 1e-4;

    % z-update with relaxation
    zold = z;
    % x-update
    x = shrinkage(zold - u, 1/rho);
    X_hat = alpha*x +(1-alpha)*zold;
    z= (X_hat+ u) + pinv(A)*(b - A*(X_hat+u));
    u = u + (X_hat - z);
    %norm of the difference between x and z
     r_norm(k)  = norm(X_hat - z);
     %norm of the difference between zold and z
     s_norm(k)  = norm(-rho*(z - zold));
     %norm of z
     obj(k)=norm(z,1)
     %if the difference is less than tolerance,then it will break the loop
     if(z-zold<RELTOL )
         g=k
         break;
     end
end
function y = shrinkage(a, kappa)
    y = max(0, a-kappa) - max(0, -a-kappa);
end
end
```

REGRESSOR SELECTION

Nonconvex problem:

$$x^* = \underset{x}{\operatorname{argmin}} \ \frac{1}{2}\|Ax - b\|_2^2$$
$$subject \ \text{to} \ card(x) \leq k$$

$$\min_{x,z} f(x) + g(z) = \frac{1}{2}\|Ax - b\|_2^2 + g(z)$$

subject to $x - z = 0$

$$g(z) = \begin{cases} 0 \text{ if } \mathrm{card}(z) \le k \\ \infty, \text{ if } \mathrm{card}(z) > k \end{cases}$$

ADMM formulation :

Augmented Lagrangian is: $L(x,z,y) = f(x) + g(z) + y^T(x - z) + \frac{\rho}{2}\|x - z\|_2^2$

Equivalent form: $L(x,z,u) = f(x) + g(z) + \frac{1}{2\lambda}\|x - z + u\|_2^2$

$$= \frac{1}{2}\|Ax - b\|_2^2 + g(z) + \frac{1}{2\lambda}\|x - z + u\|_2^2, \qquad \text{where } u = y/\rho$$

Update x:

$$x^{k+1} = \arg\min_x \frac{1}{2}\|Ax - b\|_2^2 + \frac{1}{2\lambda}\|x - z^k + u^k\|_2^2$$

Solution: Let $\rho = \frac{1}{\lambda}$

$A^T(Ax - b) + \rho(x - z^k + u^k) = 0$ vector

$(A^T A + \rho I)x = A^T b + \rho(z^k - u^k)$

$x^{k+1} = (A^T A + \rho I)^{-1}(A^T b + \rho(z^k - u^k))$

Update z:

$L(z) = g(z) + \frac{\rho}{2}\|x^{k+1} - z + u^k\|_2^2$

From the term $\|x^{k+1} - z + u^k\|_2^2$, we get z and then project (select k largest and put all other to zero)to make indicator function g(z)=0

$z^{k+1} = keep\_l\arg est(x^{k+1} + u^k)$

In Matlab:

```
function z = keep_largest(z, K)
   [val pos] = sort(abs(z), 'descend');
   z(pos(K+1:end)) = 0;
end
```

u update:

The original lagrangian multiplier term is $u^T(x-z)$

$\therefore$ The gradient w.r.t.u is $x-z$

$\Rightarrow u^{k+1} = u^k + \left(x^{k+1} - z^{k+1}\right)$

Boyd's Implementation

NB: For small dataset, matrix decomposition is not needed.

Plot:

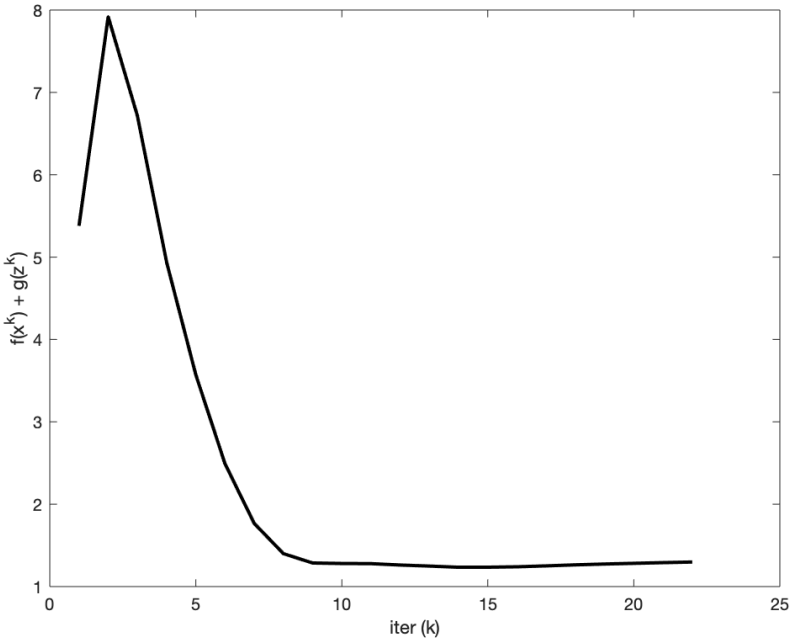


FIGURE 4

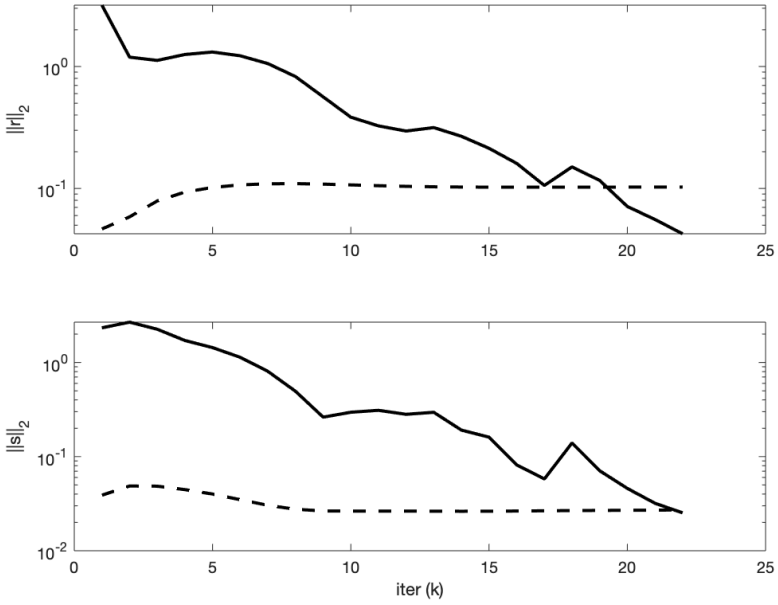


FIGURE 5

Code:

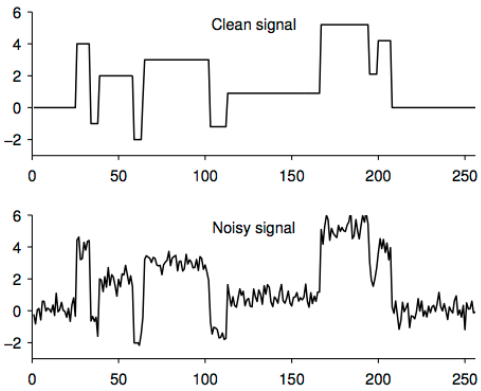
```
clc;
clear all;
close all;

randn('seed', 0);
rand('seed',0);

m = 1500;       % number of examples
n = 5000;       % number of features
p = 100/n;      % sparsity density

% generate sparse solution vector
x = sprandn(n,1,p);

% generate random data matrix
A = randn(m,n);

% normalize columns of A
A = A*spdiags(1./sqrt(sum(A.^2))', 0, n, n);

% generate measurement b with noise
b = A*x + sqrt(0.001)*randn(m,1);

xtrue = x;   % save solution

[z objval r_norm s_norm eps_pri eps_dual ] = regressor_sel(A, b, p*n, 1.0);

K = length(objval);

h = figure;
plot(1:K, objval, 'k', 'MarkerSize', 10, 'LineWidth', 2);
ylabel('f(x^k) + g(z^k)'); xlabel('iter (k)');

g = figure;
subplot(2,1,1);
semilogy(1:K, max(1e-8, r_norm), 'k', ...
    1:K, eps_pri, 'k--',  'LineWidth', 2);
ylabel('||r||_2');

subplot(2,1,2);
semilogy(1:K, max(1e-8, s_norm), 'k', ...
    1:K, eps_dual, 'k--', 'LineWidth', 2);
ylabel('||s||_2'); xlabel('iter (k)');


function [z, objval,r_norm,s_norm,eps_pri,eps_dual] = regressor_sel(A, b,
K, rho)
% regressor_sel  Solve lasso problem via ADMM
%
% [x, history] = regressor_sel(A, b, K, rho, alpha)
%
% Attempts to solve the following problem via ADMM:
%
%   minimize || Ax - b ||_2^2
%   subject to card(x) <= K
%
```

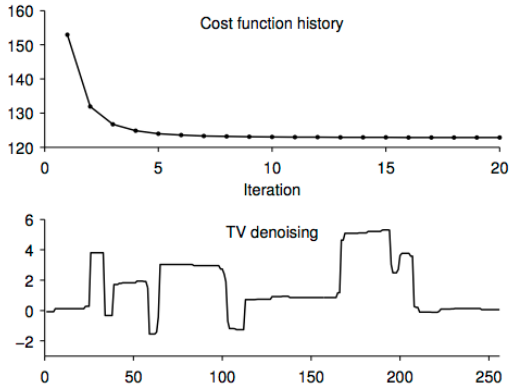
```
% where card() is the number of nonzero entries.
%
% The solution is returned in the vector x.
%
% history is a structure that contains the objective value, the primal and
% dual residual norms, and the tolerances for the primal and dual residual
% norms at each iteration.
%
% rho is the augmented Lagrangian parameter.
%
% alpha is the over-relaxation parameter (typical values for alpha are
% between 1.0 and 1.8).
%
%
% More information can be found in the paper linked at:
% http://www.stanford.edu/~boyd/papers/distr_opt_stat_learning_admm.html
%


t_start = tic;


QUIET    = 0;
MAX_ITER = 1000;
ABSTOL   = 1e-4;
RELTOL   = 1e-2;

[m, n] = size(A);

% save a matrix-vector multiply
Atb = A'*b;



x = zeros(n,1);
z = zeros(n,1);
u = zeros(n,1);

% cache the factorization
[L U] = factor(A, rho);

if ~QUIET
    fprintf('%3s\t%10s\t%10s\t%10s\t%10s\t%10s\n', 'iter', ...
      'r norm', 'eps pri', 's norm', 'eps dual', 'objective');
end

for k = 1:MAX_ITER

    % x-update
    q = Atb + rho*(z - u);    % temporary value
    if( m >= n )    % if skinny
       x = U \ (L \ q);
    else
        %if fat
       x = q/rho - (A'*(U \ ( L \ (A*q) )))/rho^2;
    end

  %  x=pinv(A'*A+rho*eye(n))*(Atb+rho*(z-u))
```

```
    % z-update with relaxation
    zold = z;
    z = keep_largest(x + u, K);

    % u-update
    u = u + (x - z);

    % diagnostics, reporting, termination checks
   objval(k)  = objective(A, b, x);

    r_norm(k)  = norm(x - z);
    s_norm(k)  = norm(-rho*(z - zold));

    eps_pri(k) = sqrt(n)*ABSTOL + RELTOL*max(norm(x), norm(-z));
    eps_dual(k)= sqrt(n)*ABSTOL + RELTOL*norm(rho*u);

    if ~QUIET
        fprintf('%3d\t%10.4f\t%10.4f\t%10.4f\t%10.4f\t%10.2f\n', k, ...
            r_norm(k), eps_pri(k), ...
            s_norm(k), eps_dual(k), objval(k));
    end

    if (r_norm(k) < eps_pri(k) && ...
       s_norm(k) < eps_dual(k))
         break;
    end

end

if ~QUIET
    toc(t_start);
end
end

function p = objective(A, b, x)
    p = sum_square(A*x - b);
end

function z = keep_largest(z, K)
    [val pos] = sort(abs(z), 'descend');
    z(pos(K+1:end)) = 0;
end

function [L U] = factor(A, rho)
    [m, n] = size(A);
    if ( m >= n )    % if skinny
       L = chol( A'*A + rho*speye(n), 'lower' );
    else            % if fat
       L = chol( speye(m) + 1/rho*(A*A'), 'lower' );
    end

    % force matlab to recognize the upper / lower triangular structure
    L = sparse(L);
    U = sparse(L');
end
```

TOTAL VARIATION MINIMIZATION

It is One of the powerful tool used in Signal and Image processing





$$\text{minimize } \frac{1}{2}\|x-b\|_2^2 + \lambda\sum_{i=1}^{n-1}|x_{i+1}-x_i|, \quad x \in R^n$$

$$\Rightarrow \text{minimize}\frac{1}{2}\|x-b\|_2^2 + \lambda\|z\|_1$$

$$Subject \text{ to } z = Dx$$

$$Augmented \text{ Lagrangian is}$$

$$L(x,z,y) = \frac{1}{2}\|x-b\|_2^2 + \lambda\|z\|_1 + y^T(Dx-z) + \frac{\rho}{2}\|Dx-z\|_2^2$$

$$\text{minimize } \frac{1}{2}\|x-b\|_2^2 + \lambda\sum_{i=1}^{n-1}|x_{i+1}-x_i|, \quad x \in R^n$$

$$\Rightarrow \text{minimize}\frac{1}{2}\|x-b\|_2^2 + \lambda\|z\|_1$$

Subject to $z = Dx$

Augmented Lagrangian is

$$L(x,z,y) = \frac{1}{2}\|x-b\|_2^2 + \lambda\|z\|_1 + y^T(Dx-z) + \frac{\rho}{2}\|Dx-z\|_2^2$$

Update x

$$L(x,z^k,y^k) = \frac{1}{2}\|x-b\|_2^2 + \left(y^k\right)^T Dx + \frac{\rho}{2}\|Dx-z^k\|_2^2$$

$$x - b + D^T y^k + \rho D^T\left(Dx - z^k\right) = 0$$

$$\left(I + \rho D^T D\right)x = b - D^T y^k + \rho D^T z^k = b + \rho D^T\left(z^k - \frac{1}{\rho}y^k\right)$$

Update z :

Augmented Lagrangian is

$$L(x,z,y) = \frac{1}{2}\|x-b\|_2^2 + \lambda\|z\|_1 + y^T(Dx-z) + \frac{\rho}{2}\|Dx-z\|_2^2$$

$$L(x^{k+1},z,y^k) = \lambda\|z\|_1 - \left(y^k\right)^T z + \frac{\rho}{2}\|Dx^{k+1}-z\|_2^2$$

$$L(x^{k+1},z,y^k) = \|z\|_1 + \frac{\rho}{2\lambda}\left\|Dx^{k+1} - z + \frac{y^k}{\rho}\right\|_2^2$$

$$z^{k+1} = S_{\lambda/\rho}\left(Dx^{k+1} + \frac{y^k}{\rho}\right)$$

Update u :

$$y^{k+1} = y^k + Dx^{k+1} - z^k$$

Three update rules are

$$x^{k+1} = \left(I + \rho D^T D\right)^{-1}\left(b + \rho D^T\left(z^k - \frac{1}{\rho}y^k\right)\right) - - - (1)$$

$$z^{k+1} = S_{\lambda/\rho}\left(Dx^{k+1} + \frac{y^k}{\rho} \right) ---------(2)$$

$$y^{k+1} = y^k + \left(Dx^{k+1} - z^k \right) ---------(3)$$

$$On \text{ letting } \frac{1}{\rho} y^k = u^k$$

$$x^{k+1} = \left(I + \rho D^T D \right)^{-1} \left(b + \rho D^T \left(z^k - u^k \right) \right) ---(1)$$

$$z^{k+1} = S_{\lambda/\rho}\left(Dx^{k+1} + u^k \right) ---------(2)$$

$$u^{k+1} = u^k + \left(Dx^{k+1} - z^k \right) ---------(3)$$

Plot:

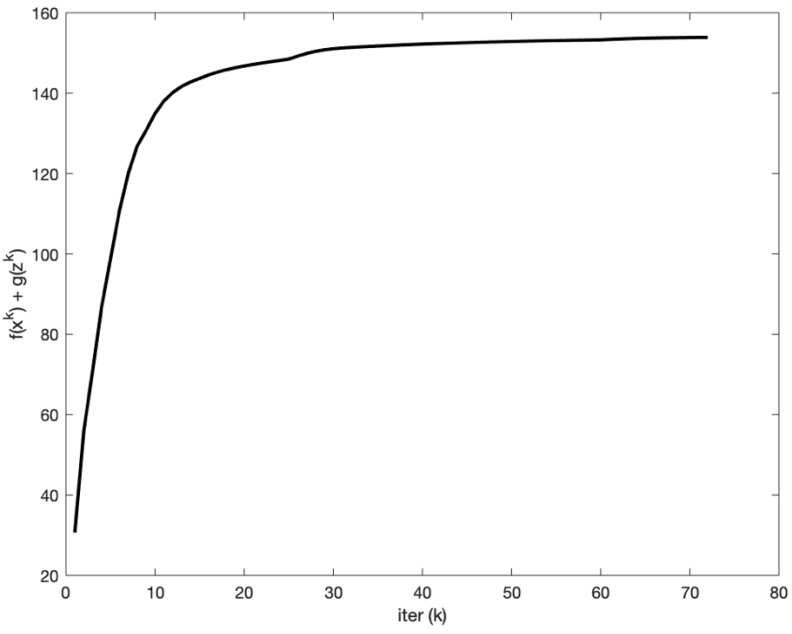


FIGURE 6

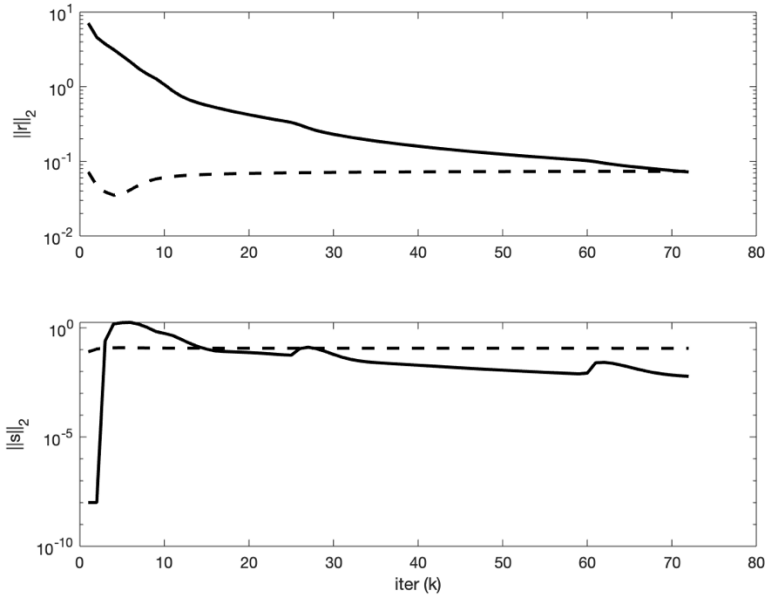


FIGURE 7

Code:

```
clc;
clear all;
close all;

rand('seed', 0);
randn('seed', 0);

n = 100;

x0 = ones(n,1);
for j = 1:3
    idx = randsample(n,1);
    k = randsample(1:10,1);
    x0(ceil(idx/2):idx) = k*x0(ceil(idx/2):idx);
end
b = x0 + randn(n,1);

lambda = 5;

e = ones(n,1);
D = spdiags([e -e], 0:1, n,n);

[x objval r_norm s_norm eps_pri eps_dual] = total_variation(b, lambda, 1.0,
1.0);
K = length(objval);

h = figure;
plot(1:K, objval, 'k', 'MarkerSize', 10, 'LineWidth', 2);
ylabel('f(x^k) + g(z^k)'); xlabel('iter (k)');

g = figure;
```

```
subplot(2,1,1);
semilogy(1:K, max(1e-8, r_norm), 'k', ...
    1:K, eps_pri, 'k--',  'LineWidth', 2);
ylabel('||r||_2');

subplot(2,1,2);
semilogy(1:K, max(1e-8, s_norm), 'k', ...
    1:K, eps_dual, 'k--', 'LineWidth', 2);
ylabel('||s||_2'); xlabel('iter (k)');
```

```
function [x, objval,r_norm,s_norm,eps_pri,eps_dual] = total_variation(b,
lambda, rho, alpha)

% total_variation  Solve total variation minimization via ADMM
%
% [x, history] = total_variation(b, lambda, rho, alpha)
%
% Solves the following problem via ADMM:
%
%   minimize  (1/2)||x - b||_2^2 + lambda * sum_i |x_{i+1} - x_i|
%
% where b in R^n.
%
% The solution is returned in the vector x.
%
% history is a structure that contains the objective value, the primal and
% dual residual norms, and the tolerances for the primal and dual residual
% norms at each iteration.
%
% rho is the augmented Lagrangian parameter.
%
% alpha is the over-relaxation parameter (typical values for alpha are
% between 1.0 and 1.8).
%
%
% More information can be found in the paper linked at:
% http://www.stanford.edu/~boyd/papers/distr_opt_stat_learning_admm.html
%

t_start = tic;
QUIET    = 0;
MAX_ITER = 1000;
ABSTOL   = 1e-4;
RELTOL   = 1e-2;


n = length(b);

% difference matrix
e = ones(n,1);
D = spdiags([e -e], 0:1, n,n);
x = zeros(n,1);
z = zeros(n,1);
u = zeros(n,1);

if ~QUIET
```

```
    fprintf('%3s\t%10s\t%10s\t%10s\t%10s\t%10s\n', 'iter', ...
      'r norm', 'eps pri', 's norm', 'eps dual', 'objective');
end

I = speye(n);
DtD = D'*D;

for k = 1:MAX_ITER

    % x-update
    x = (I + rho*DtD) \ (b + rho*D'*(z-u));

    % z-update with relaxation
    zold = z;
    Ax_hat = alpha*D*x +(1-alpha)*zold;
    z = shrinkage(Ax_hat + u, lambda/rho);

    % y-update
    u = u + Ax_hat - z;


    % diagnostics, reporting, termination checks
    objval(k)  = objective(b, lambda, D, x, z);

    r_norm(k)  = norm(D*x - z);
    s_norm(k)  = norm(-rho*D'*(z - zold));

    eps_pri(k) = sqrt(n)*ABSTOL + RELTOL*max(norm(D*x), norm(-z));
    eps_dual(k)= sqrt(n)*ABSTOL + RELTOL*norm(rho*D'*u);

    if ~QUIET
        fprintf('%3d\t%10.4f\t%10.4f\t%10.4f\t%10.4f\t%10.2f\n', k, ...
            r_norm(k), eps_pri(k), ...
            s_norm(k), eps_dual(k),objval(k));
    end

    if (r_norm(k) < eps_pri(k) && ...
       s_norm(k) < eps_dual(k))
         break;
    end
end

if ~QUIET
    toc(t_start);
end
end

function obj = objective(b, lambda, D, x, z)
    obj = .5*norm(x - b)^2 + lambda*norm(z,1);
end

function y = shrinkage(a, kappa)
    y = max(0, a-kappa) - max(0, -a-kappa);
end
```

LASSO

(LEAST ABSOLUTE SHRINKAGE AND SELECTION OPERATION)

LASSO (Least Absolute Shrinkage and Selection Operator)
The following L_1 regularized least-squares problem :

$$\underset{x}{\text{minnimize}} \frac{1}{2}\|Ax-b\|_2^2 + \tau\|x\|_1 \;;\; \left(\text{or} \underset{x}{\text{minnimize}} \;\frac{1}{2}(Ax-b)^T(Ax-b) + \tau\sum_i |x_i|\right)$$

is called the LASSO.
It is prevalent all across machine learning, model selection in statistics, and compressed sensing in signal processing.
The $\tau > 0$ above is a user-defined "*smoothing* parameter"

Taking $f(x) = \frac{1}{2}\|Ax-b\|_2^2$ and g(z)= $\tau\|z\|_1$
ADMM formulation is:
$\underset{x,z}{\text{minimize}}$ $f(x) + g(z)$ subject to $x - z = \mathbf{0}$ vector

Augmented Lagrangian is

$$L(x,z,y) = f(x) + g(z) + y^T(x-z) + \frac{\rho}{2}(x-z)^T(x-z)$$

$$L(x,z,y) = \frac{1}{2}(Ax-b)^T(Ax-b) + \tau\|z\|_1 + y^T(x-z) + \frac{\rho}{2}(x-z)^T(x-z)$$

Re*duced* equivalent form is (*y is replaced by equivalent u*)

$$L(x,z,u) = \frac{1}{2}(Ax-b)^T(Ax-b) + \tau\|z\|_1 + \frac{1}{2\lambda}\|x-z+u\|_2^2$$

$$x^{(k+1)} = \arg\min_x \frac{1}{2}(Ax-b)^T(Ax-b) + \frac{1}{2\lambda}\|x-z^k+u^k\|_2^2$$

$$x^{(k+1)} = \arg\min_x \frac{1}{2}\left(Ax-b\right)^T\left(Ax-b\right) + \frac{1}{2\lambda}\left\|x - z^k + u^k\right\|_2^2$$

on differentiating and putting equal to 0 vector

$$A^T(Ax-b) + \frac{1}{\lambda}\left(x - z^k + u^k\right) = \mathbf{0} \text{ vector}$$

$$A^TAx + \frac{1}{\lambda}x = A^Tb + \frac{1}{\lambda}\left(z^k - u^k\right)$$

$$\left(A^TA + \frac{1}{\lambda}I\right)x = A^Tb + \frac{1}{\lambda}\left(z^k - u^k\right)$$

$$x = x^{k+1} = \left(A^TA + \frac{1}{\lambda}I\right)^{-1}\left(A^Tb + \frac{1}{\lambda}\left(z^k - u^k\right)\right)$$

update of z is:

$$L(x,z,y) = \frac{1}{2}\left(Ax-b\right)^T\left(Ax-b\right) + \tau\left\|z\right\|_1 + \frac{1}{2\lambda}\left\|x - z + u\right\|_2^2$$

$$z^{(k+1)} = \arg\min_z \tau\left\|z\right\|_1 + \frac{1}{2\lambda}\left\|x^{k+1} - z + u^k\right\|_2^2 \triangleq \arg\min_z \left\|z\right\|_1 + \frac{1}{2\lambda\tau}\left\|x^{k+1} - z + u^k\right\|_2^2$$

$$z^{(k+1)} = S_{\lambda\tau}\left(x^{k+1} + u^k\right)$$

REMEMBER:

$$\min_x f(x) = \left\|x\right\|_1 + \frac{1}{2\lambda}\left\|x-c\right\|^2$$

$$sign(x) + \frac{1}{\lambda}(x-c) = 0 \Rightarrow x^* = c - \lambda \times sign(c)$$

$$x^* = S_\lambda(c) = \begin{cases} c - \lambda \times sign(c), & \text{if } \lambda \le |c|, \\ 0, & \text{if } \lambda > |c| \end{cases}$$

To Summarise

$$x^{k+1} = \left(A^TA + \frac{1}{\lambda}I\right)^{-1}\left(A^Tb + \frac{1}{\lambda}\left(z^k - u^k\right)\right)$$

$$z^{k+1} = S_{\lambda\tau}(x^{k+1} + u^k)$$

$$u^{k+1} = u^k + (x^{k+1} - z^{k+1})$$

Plot:

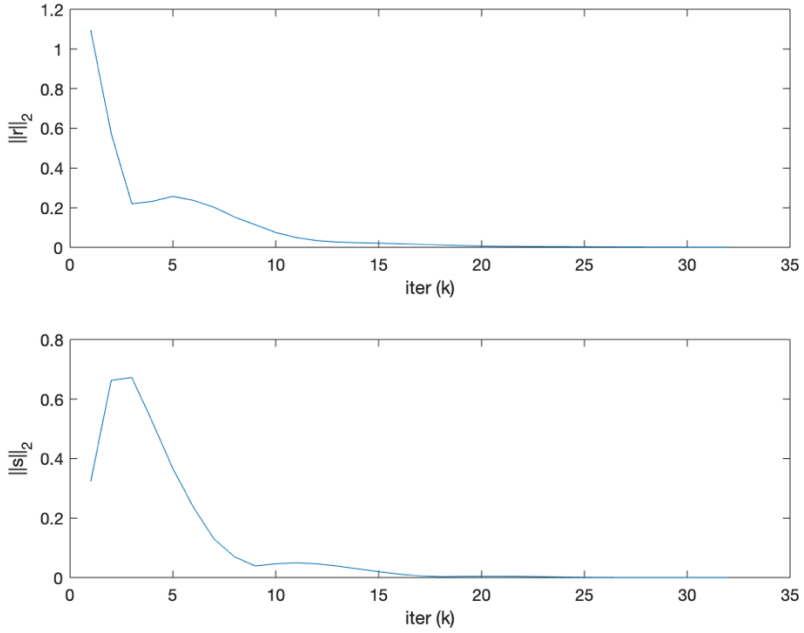


FIGURE 8

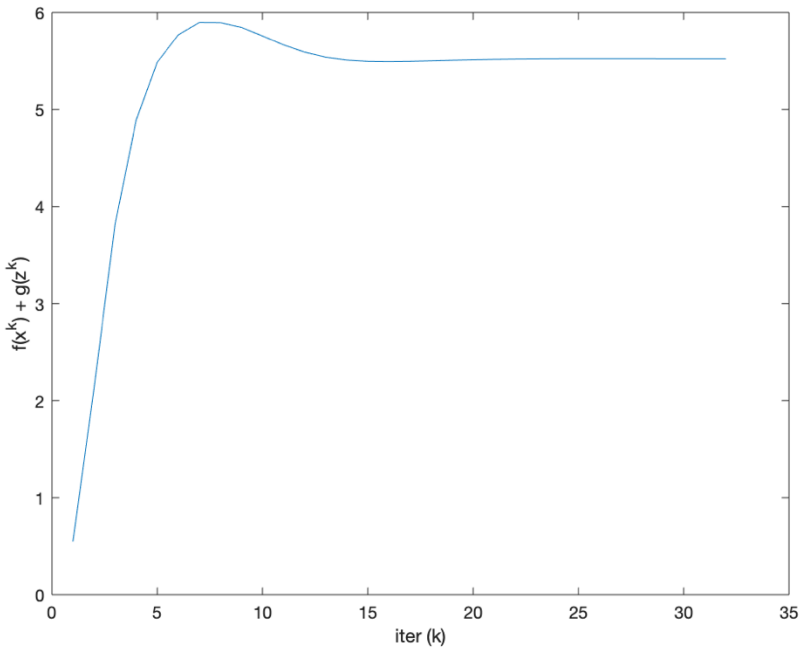


FIGURE 9

Code:

```
% lasso  Solve lasso problem via ADMM

% [z, history] = lasso(A, b, lambda, rho, alpha);
% Solves the following problem via ADMM:
%   minimize 1/2*|| Ax - b ||_2^2 + \lambda || x ||_1
% The solution is returned in the vector x.
% rho is the augmented Lagrangian parameter.
% alpha is the over-relaxation parameter (typical values for alpha are
% between 1.0 and 1.8).

randn('seed', 0);
rand('seed',0);
m = 150;        % number of examples
n = 500;        % number of features
p = 10/n;       % sparsity density
x0 = sprandn(n,1,p);
A = randn(m,n);
A = A*spdiags(1./sqrt(sum(A.^2))',0,n,n); % normalize columns
b = A*x0 + sqrt(0.001)*randn(m,1);

lambda_max = norm( A'*b, 'inf' );
lambda = 0.1*lambda_max;
%calling basis_pusuit function
[x z u r_norm s_norm g obj] = lasso(A, b, lambda, 1.0, 1.0);
% for printing  'x^(k+1)','z^(k+1)','u^(k+1)'
fprintf('%20s\t%10s\t%10s\t \n', 'x^(k+1)','z^(k+1)','u^(k+1)');
for i =1:length(x)
fprintf('%20f\t%10.4f\t%10.4f\t \n', x(i),z(i),u(i));
end
figure
%plotting difference between current z and old z
subplot(2,1,1)
plot(1:g,r_norm)
ylabel('||r||_2'); xlabel('iter (k)');
%plotting difference between z and x
subplot(2,1,2)
plot(1:g,s_norm)
ylabel('||s||_2'); xlabel('iter (k)');
figure
%plotting norm with iteration
plot(1:g,obj)
ylabel('f(x^k) + g(z^k)'); xlabel('iter (k)');
function [X_hat z u r_norm s_norm g obj] = lasso(A, b,lambda, rho, alpha)
MAX_ITER = 100;
[m n] = size(A);
x = zeros(n,1);
z = zeros(n,1);
u = zeros(n,1);
for k = 1:MAX_ITER
    %giving tolerance value
    RELTOL   = 1e-4;

    % z-update with relaxation
    zold = z;
    % x-update
```

```
    x= pinv(A'*A+(rho)*speye(n)) *( A'*b+(rho)*(zold-u));
    X_hat = alpha*x +(1-alpha)*zold;
    z = shrinkage(X_hat + u,lambda);
    u = u + (X_hat - z);
    %norm of the difference between x and z
     r_norm(k)  = norm(X_hat - z);
     %norm of the difference between zold and z
     s_norm(k)  = norm(-rho*(z - zold));
     %norm of z
     obj(k)=norm(z,1)
     %if the difference is less than tolerance
     g=k
     if(z-zold<RELTOL )
         g=k
         break;
     end
end
function y = shrinkage(a, kappa)
    y = max(0, a-kappa) - max(0, -a-kappa);
end
end
```

HUBER LOSS FUNCTION

The typical loss functions are

Least Square (L_2 norm)

Least absolute value (min L_1 norm)

Hinge loss (one sided L_1 norm)

Epsino insensitive loss function

Huber loss function:

$Let\ x \in R^n$

$z_{m\times 1} = A_{m\times n}x - b_{m\times 1}$

$z_i = \left(A_{m\times n}x\right)_i - b_i;\ \ i=1,2,..,m$

Given matrix A and vector b, find x such that

$\sum_{i=1}^{m} huber(z_i)$ is minimum

where

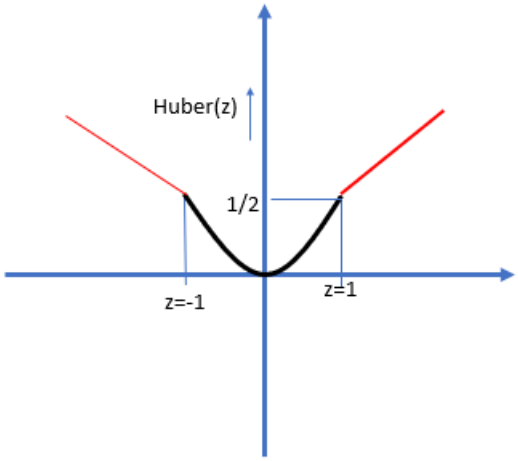
$$huber(z_i) = \frac{1}{2}z_i^2, \quad |z_i| \le 1$$

$$huber(z_i) = |z_i| - \frac{1}{2}, \quad |z_i| > 1$$

The optimization problem is

$$\min_{x,z} \sum_{i=1}^{m} huber(z_i)$$

$$Ax - z = b$$



For deviation from target in the range -1 to 1 is limited to

0 to ½ (squared loss). Beyond that, loss is increases linearly

with deviation

$$\min_{x,z} \; g(z) \; = \; \sum_{i=1}^{m} huber(z_i)$$

$$subject \text{ to } z = Ax - b$$

$$ADMM \text{ formulation is}$$

$$\min_{x,z} g(z)$$

$$subject \text{ to } z = Ax - b$$

$$\Rightarrow Ax - b - z = \mathbf{0} \text{ vector}$$

$$Augmented \text{ Lagrangian is}$$

$$L(x,z,u) = g(z) + y^T (Ax - b - z) + \frac{\rho}{2} \| Ax - b - z \|_2^2$$

$$Equivalent \text{ simplified form is}$$

$$L(x,z,u) = g(z) + \frac{1}{2\lambda} \| Ax - b - z + u \|_2^2$$

Update x:

$$\text{L(x,z,u)}=g(z)+\frac{1}{2\lambda}\left\|Ax-b-z+u\right\|_2^2$$

$$x^{k+1}=\arg\min_x\left\|Ax-b-z^k+u^k\right\|_2^2$$

Solution:

$$Let\ f(x)=\left\|Ax-b-z^k+u^k\right\|_2^2$$

$$=\left(Ax-b-z^k+u^k\right)^T\left(Ax-b-z^k+u^k\right)$$

$$\nabla f(x)=\mathbf{0}\Rightarrow A^T\left(Ax-b-z^k+u^k\right)=\mathbf{0}$$

$$A^TAx=A^T\left(b+z^k-u^k\right)$$

$$x^{k+1}=\left(A^TA\right)^{-1}A^T\left(b+z^k-u^k\right)$$

Boyd used Cholesky decomposition here

Update z:

$$\text{L(x,z,u)}=g(z)+\frac{1}{2\lambda}\left\|Ax-b-z+u\right\|_2^2$$

$$z^{k+1}=\arg\min_z g(z)+\frac{1}{2\lambda}\left\|Ax^{k+1}-b-z+u^k\right\|_2^2$$

Solution: g(z)=Huber(z) consists of two piecewise continous functions .
Let us consider first one.

$$Let\ g(z)=\frac{z^2}{2},\ \text{then}$$

$$z^{k+1}=\arg\min_z\frac{z^2}{2}+\frac{1}{2\lambda}\left\|Ax^{k+1}-b-z+u^k\right\|_2^2$$

$$z-\frac{1}{\lambda}\left(Ax^{k+1}-b-z+u^k\right)=0\Rightarrow(1+\lambda)z^{k+1}=\left(Ax^{k+1}-b+u^k\right)$$

$$z_i^{k+1}=\frac{1}{1+\lambda}\left(Ax^{k+1}-b+u^k\right)_i\text{ ----------(1)}$$

$\left(Ax^{k+1}-b+u^k\right)_i$ is the ith element of vector $\left(Ax^{k+1}-b+u^k\right)$

Update z: Consider second function

Let $g(z) = \|z\|_1 - \frac{1}{2}$, then

$$z^{k+1} = \arg\min_z \|z\|_1 - \frac{1}{2} + \frac{1}{2\lambda}\left\|Ax^{k+1} - b - z + u^k\right\|_2^2$$

$$z^{k+1} = \arg\min_z \|z\|_1 + \frac{1}{2\lambda}\left\|Ax^{k+1} - b - z + u^k\right\|_2^2$$

$$z_i^{k+1} = S_\lambda\left(\left(Ax^{k+1} - b + u^k\right)_i\right) = \left(Ax^{k+1} - b + u^k\right)_i \pm \lambda \text{ ---------(2)}$$

S_λ represent shrinkage by $\pm\lambda$, sign depending on the sign of $\left(Ax^{k+1} - b + u^k\right)_i$

$\left(Ax^{k+1} - b + u^k\right)_i$ is the ith element of vector $\left(Ax^{k+1} - b + u^k\right)$

How will we combine the two formula (1) and (2) for z update?.

Split z_i^{k+1} *in* (2) int*o* two terms, one term containing (1) and express the residual in terms of a Shrinkage function

$$z_i^{k+1} = \left(Ax^{k+1} - b + u^k\right)_i \pm \lambda ----(2)$$

$$z_i^{k+1} = \left(Ax^{k+1} - b + u^k\right)_i \pm \lambda = \frac{1}{1+\lambda}\left(Ax^{k+1} - b + u^k\right)_i + \frac{\lambda}{1+\lambda}\left\{\left(Ax^{k+1} - b + u^k\right)_i \pm (1+\lambda)\right\}$$

$$= \frac{1}{1+\lambda}\left(Ax^{k+1} - b + u^k\right)_i + \frac{\lambda}{1+\lambda} S_{1+\lambda}\left(Ax^{k+1} - b + u^k\right)_i$$

or

$$z^{k+1} = \frac{1}{1+\lambda}\left(Ax^{k+1} - b + u^k\right) + \frac{\lambda}{1+\lambda} S_{1+\lambda}\left(Ax^{k+1} - b + u^k\right)$$

u update:

The original lagrangian multiplier term is $u^T(Ax - z - b)$

$\therefore$ The gradient is $Ax - z - b$

$$\Rightarrow u^{k+1} = u^k + \left(Ax^{k+1} - z^{k+1} - b\right)$$

Plots:

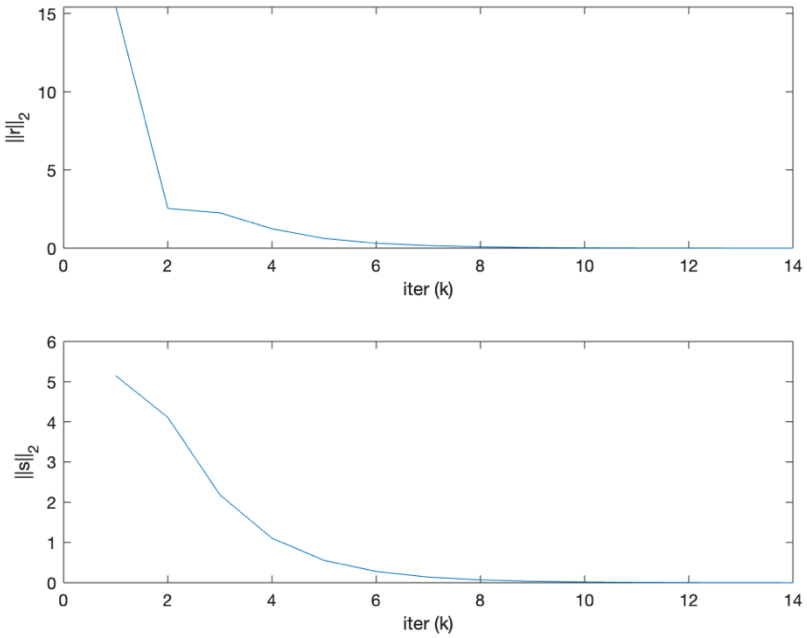


FIGURE 10

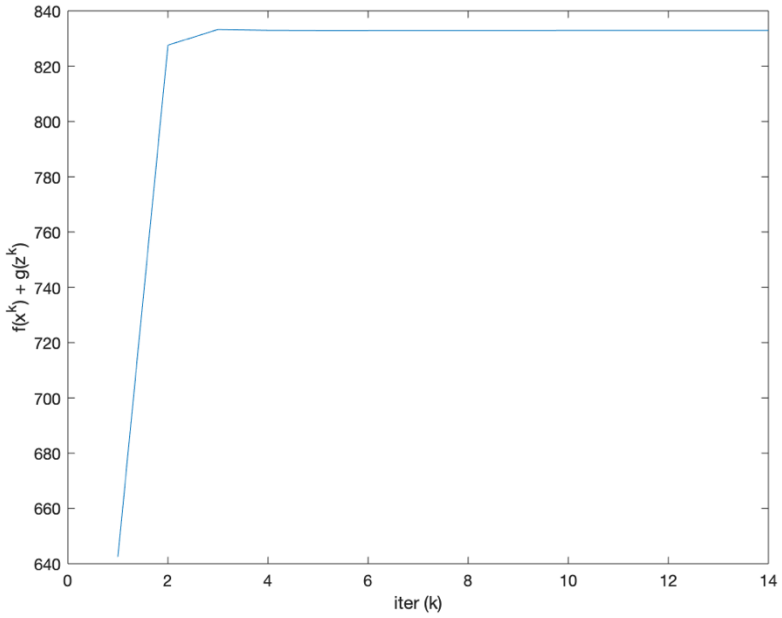


FIGURE 11

Code:

```
% basis_pursuit  Solve basis pursuit via ADMM
%
% [x, history] = basis_pursuit(A, b, rho, alpha)
%
% Solves the following problem via ADMM:
%
%   minimize     ||x||_1
%   subject to   Ax = b
%
% The solution is returned in the vector x.
% rho is the augmented Lagrangian parameter.
randn('seed', 0);
rand('seed',0);

m = 5000;       % number of examples
n = 200;        % number of features

x0 = randn(n,1);
A = randn(m,n);
A = A*spdiags(1./norms(A)',0,n,n); % normalize columns
b = A*x0 + sqrt(0.01)*randn(m,1);
b = b + 10*sprand(m,1,200/m);
%calling basis_pusuit function
[x z u r_norm s_norm g obj] = lasso(A, b, 1.0, 1.0);
% for printing  'x^(k+1)','z^(k+1)','u^(k+1)'
fprintf('%20s\t%10s\t%10s\t \n', 'x^(k+1)','z^(k+1)','u^(k+1)');
for i =1:length(x)
fprintf('%20f\t%10.4f\t%10.4f\t \n', x(i),z(i),u(i));
end
figure
%plotting difference between current z and old z
subplot(2,1,1)
plot(1:g,r_norm)
ylabel('||r||_2'); xlabel('iter (k)');
%plotting difference between z and x
subplot(2,1,2)
plot(1:g,s_norm)
ylabel('||s||_2'); xlabel('iter (k)');
figure
%plotting norm with iteration
plot(1:g,obj)
ylabel('f(x^k) + g(z^k)'); xlabel('iter (k)');
function [Ax_hat z u r_norm s_norm g obj] = lasso(A, b, rho, alpha)
QUIET    = 0;
MAX_ITER = 1000;
ABSTOL   = 1e-4;
RELTOL   = 1e-2;
[m, n] = size(A);
Atb = A'*b;
[L U] = factor(A);
[m n] = size(A);
x = zeros(n,1);
z = zeros(m,1);
u = zeros(m,1);
for k = 1:MAX_ITER
    %giving tolerance value
    RELTOL   = 1e-4;
```

```
    % x-update
     q = Atb + A'*(z - u);
     x = U \ (L \ q);

     % z-update with relaxation
     zold = z;
     Ax_hat = alpha*A*x + (1-alpha)*(zold + b);
     tmp = Ax_hat - b + u;
     z = rho/(1 + rho)*tmp + 1/(1 + rho)*shrinkage(tmp, 1 + 1/rho);

     u = u + (Ax_hat - z - b);
     % diagnostics, reporting, termination checks
     obj(k)  = ( 1/2*sum(huber(z)) );;
     r_norm(k)  = norm(A*x - z - b);
     s_norm(k)  = norm(-rho*A'*(z - zold));

      g=k
      if(z-zold<RELTOL )
          g=k
          break;
      end
end
function z = shrinkage(x, kappa)
    z = pos(1 - kappa./abs(x)).*x;
end

function [L U] = factor(A)
    [m, n] = size(A);
    if ( m >= n )    % if skinny
       L = chol( A'*A, 'lower' );
    end

    % force matlab to recognize the upper / lower triangular structure
    L = sparse(L);
    U = sparse(L');
end
end
```

Least absolute deviations

We now look at similar problem(Basis Pursuit) of minimizing in the L1 sense.

Least Absolute Deviations (LAD) is denoted by the problem of minimizing the objective:

$$x^* = \underset{x}{\operatorname{argmin}} \|Ax - b\|_1 \quad , \qquad \left(\|Ax - b\|_1 \equiv \sum_{i=1}^{n} \left| a_i^T x - b_i \right|, \quad a_i^T \text{ is ith row of A} \right)$$

This problem differs slightly from Basis Pursuit in that we do not have the constraint Ax = b anymore. We are given observation in the rows of A and corresponding noisy measurements b. Our goal here is to actually recover the original, minimal signal x,not denoise it, despite the noise in b .

$$Let\ z\ =\ Ax - b$$

$$ADMM\ \text{ formulation is}$$

$$\min_{x,z} \|z\|_1$$

$$subject\ \text{to}\ z = Ax - b \Rightarrow Ax - b - z = \mathbf{0}\ \text{vector}$$

$$Augmented\ \text{Lagrangian is}$$

$$\text{L(x,z,u)} = \|z\|_1 + \frac{1}{2\lambda} \|Ax - b - z + u\|_2^2$$

This part of formulation is tricky.

Depending on problem, mathematicians innovatively come with different formulations

Update x:

$$L(x,z,u)=\|z\|_1+\frac{1}{2\lambda}\|Ax-b-z+u\|_2^2$$

$$x^{k+1}=\arg\min_x \left\|Ax-b-z^k+u^k\right\|_2^2$$

Solution:

$$Let\ f(x)=\left\|Ax-b-z^k+u^k\right\|_2^2=\left(Ax-b-z^k+u^k\right)^T\left(Ax-b-z^k+u^k\right)$$

$$\nabla f(x)=\mathbf{0}\Rightarrow A^T\left(Ax-b-z^k+u^k\right)=\mathbf{0}$$

$$A^TAx=A^T\left(b+z^k-u^k\right)$$

$$x^{k+1}=\left(A^TA\right)^{-1}A^T\left(b+z^k-u^k\right)$$

Instead of naively solving this system, we use a more efficient approach by finding the Cholesky decomposition $RR^T = A^TA$, where R and R^T are lower and upper triangular, respectively. We perform two easy system solves (due to R's triangularity) by ding a y such that $Ry = A^T(b+z^k-u^k)$ and then finding our minimizing x such that $R^Tx = y$.

Update z:

$$\mathrm{L(x,z,u)}=\|z\|_1+\frac{1}{2\lambda}\|Ax-b-z+u\|_2^2$$

$$z^{k+1}=\arg\min_z \|z\|_1+\frac{1}{2\lambda}\left\|Ax^{k+1}-b-z+u^k\right\|_2^2$$

Solution:

$z^{k+1}=S_\lambda\left(Ax^{k+1}-b+u^k\right)$; S_λ is elementwise shrinkage operation by λ towards zero

Boyd applied relaxation here

Update z:

$$\mathrm{L(x,z,u)}=\|z\|_1+\frac{1}{2\lambda}\|Ax-b-z+u\|_2^2$$

$$A\hat{x}=\alpha Ax^{k+1}+(1-\alpha)\underbrace{\left(z^k+b\right)}_{an\ \text{estimate of Ax}}$$

$z^{k+1}=S_\lambda\left(A\hat{x}-b+u^k\right)$; S_λ is elementwise shrinkage operation by λ towards zero

Update u:

$$\mathrm{L(x,z,u)}=\|z\|_1+\frac{1}{2\lambda}\|Ax-b-z+u\|_2^2$$

$\frac{\partial L}{\partial u}=Ax^{k+1}-b-z^{k+1}\equiv gradient$. This expression is from lagangian

$u^{k+1}=u^k+\left(Ax^{k+1}-b-z^{k+1}\right)$

Boyd did a change here

Update u:

$u^{k+1}=u^k+\left(A\hat{x}-b-z^{k+1}\right)$

Plot:

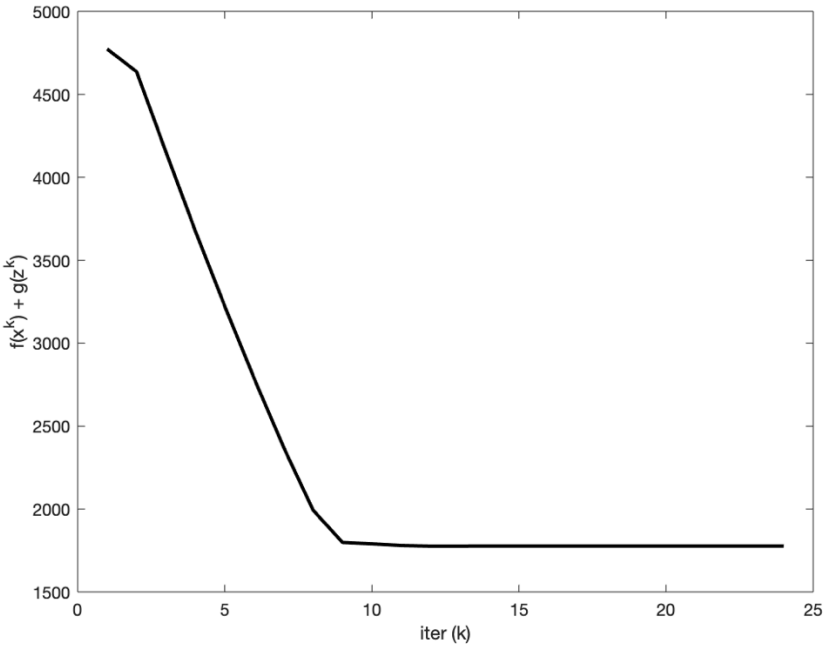


FIGURE 12

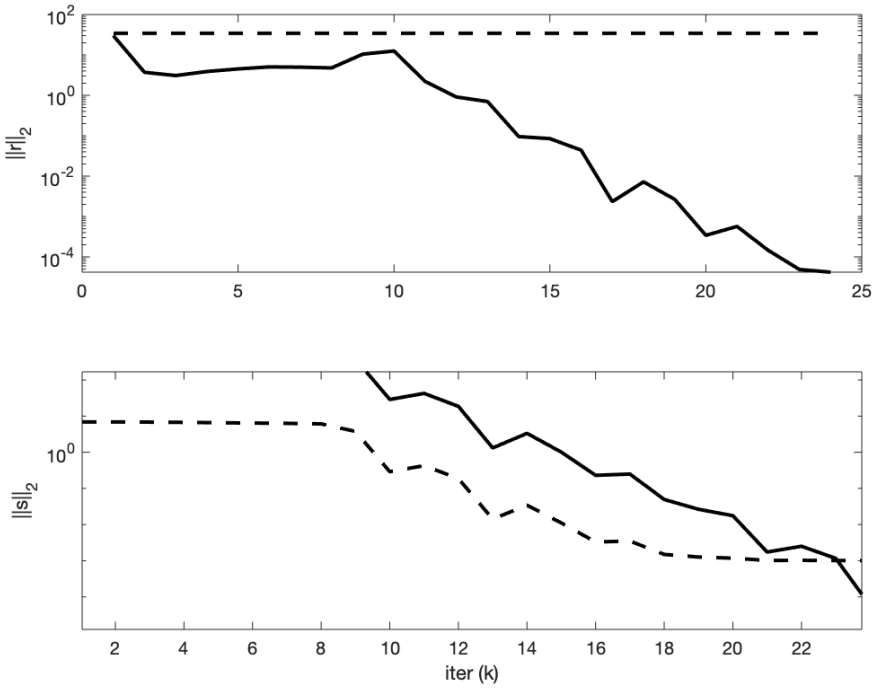


FIGURE 13

Code:

```
clc;
clear all;
close all;

rand('seed', 0);
randn('seed', 0);

m = 1000; % number of examples
n = 100;  % number of features

A = randn(m,n);
x0 = 10*randn(n,1);
b = A*x0;
idx = randsample(m,ceil(m/50));
b(idx) = b(idx) + 1e2*randn(size(idx));

[x objval r_norm s_norm eps_pri eps_dual] = lad(A, b, 1.0, 1.0);

K = length(objval);

h = figure;
plot(1:K, objval, 'k', 'MarkerSize', 10, 'LineWidth', 2);
ylabel('f(x^k) + g(z^k)'); xlabel('iter (k)');

g = figure;
subplot(2,1,1);
```

```
semilogy(1:K, max(1e-8,r_norm), 'k', ...
    1:K, eps_pri, 'k--',  'LineWidth', 2);
ylabel('||r||_2');

subplot(2,1,2);
semilogy(1:K, max(1e-8, s_norm), 'k', ...
    1:K, eps_dual, 'k--', 'LineWidth', 2);
ylabel('||s||_2'); xlabel('iter (k)');



function [x,objval,r_norm,s_norm,eps_pri,eps_dual] = lad(A, b, rho, alpha)
% lad  Least absolute deviations fitting via ADMM
%
% [x, history] = lad(A, b, rho, alpha)
%
% Solves the following problem via ADMM:
%
%   minimize     ||Ax - b||_1
%
% The solution is returned in the vector x.
%
% history is a structure that contains the objective value, the primal and
% dual residual norms, and the tolerances for the primal and dual residual
% norms at each iteration.
%
% rho is the augmented Lagrangian parameter.
%
% alpha is the over-relaxation parameter (typical values for alpha are
% between 1.0 and 1.8).
%
%
% More information can be found in the paper linked at:
% http://www.stanford.edu/~boyd/papers/distr_opt_stat_learning_admm.html
%

t_start = tic;

QUIET    = 0;
MAX_ITER = 1000;
ABSTOL   = 1e-4;
RELTOL   = 1e-2;

[m n] = size(A);
x = zeros(n,1);
z = zeros(m,1);
u = zeros(m,1);

if ~QUIET
    fprintf('%3s\t%10s\t%10s\t%10s\t%10s\t%10s\n', 'iter', ...
      'r norm', 'eps pri', 's norm', 'eps dual', 'objective');
end

for k = 1:MAX_ITER

    if k > 1
        x = R \ (R' \ (A'*(b + z - u)));
    else
        R = chol(A'*A);
```

```
        x = R \ (R' \ (A'*(b + z - u)));
    end

    zold = z;
    Ax_hat = alpha*A*x + (1-alpha)*(zold + b);
    z = shrinkage(Ax_hat - b + u, 1/rho);

    u = u + (Ax_hat - z - b);

    % diagnostics, reporting, termination checks

    objval(k)  = objective(z);

    r_norm(k)  = norm(A*x - z - b);
    s_norm(k)  = norm(-rho*A'*(z - zold));

    eps_pri(k) = sqrt(m)*ABSTOL + RELTOL*max([norm(A*x), norm(-z),
norm(b)]);
   eps_dual(k)= sqrt(n)*ABSTOL + RELTOL*norm(rho*A'*u);

    if ~QUIET
        fprintf('%3d\t%10.4f\t%10.4f\t%10.4f\t%10.4f\t%10.2f\n', k, ...
            r_norm(k), eps_pri(k), ...
          s_norm(k), eps_dual(k), objval(k));
    end

    if (r_norm(k) < eps_pri(k) && ...
         s_norm(k) < eps_dual(k))
         break;
    end
end

if ~QUIET
    toc(t_start);
end
end

function obj = objective(z)
    obj = norm(z,1);
end

function y = shrinkage(a, kappa)
    y = max(0, a-kappa) - max(0, -a-kappa);
end
```

Quadratic programming

$$\min \quad \left(\frac{1}{2}\right) x^T Px + q^T x$$
$$subject\ \ to\ Ax = b, x \geq 0$$

Reduce this to

$$\min \quad f(x) + g(z)$$
$$subject\ \ to\ \ x - z = 0$$

$$f(x) = \left(\frac{1}{2}\right) x^T P x + q^T x, \quad domf\{x \mid Ax = b,\}$$

$g(z)$, Indicator Function for $z \geq 0$

$$L_\rho(x,z,u) = f(x) + g(z) + \left(\frac{1}{2\lambda}\right)\|x - z + u\|_2^2$$

$$x^{k+1} = \underset{x:Ax=b}{\arg\min}\left(f(x) + \frac{1}{2\lambda}\left\|x - z^k + u^k\right\|_2^2 \right)$$

$$L(x,v) = \left(\frac{1}{2}\right) x^T P x + q^T x + \frac{1}{2\lambda}\left\|x - z^k + u^k\right\|_2^2 + v^T(Ax - b)$$

$$\frac{\partial L}{\partial x} = Px + q + \frac{1}{\lambda}\left(x - z^k + u^k\right) + A^T v$$

$$\frac{\partial L}{\partial v} = Ax - b$$

$$\frac{\partial L}{\partial x} = 0 \Rightarrow Px^{k+1} + q + \frac{1}{\lambda}\left(x^{k+1} - z^k + u^k\right) + A^T v = 0$$

$$Px^{k+1} + q + \frac{1}{\lambda}\left(x^{k+1} - z^k + u^k\right) + A^T v = 0$$

$$Px^{k+1} + q + \frac{1}{\lambda}x^{k+1} - \frac{1}{\lambda}z^k + \frac{1}{\lambda}u^k + A^T v = 0$$

$$\left(P + \frac{1}{\lambda}I\right)x^{k+1} + q + A^T v - \frac{1}{\lambda}\left(z^k - u^k\right) = 0$$

$$\frac{\partial L}{\partial v} = 0 \Rightarrow Ax^{k+1} - b = 0$$

$$\begin{bmatrix} P+\frac{1}{\lambda}I & A^T \\ A & 0 \end{bmatrix}\begin{bmatrix} x^{k+1} \\ v \end{bmatrix}+\begin{bmatrix} q-\frac{1}{\lambda}(z^k-u^k) \\ -b \end{bmatrix}=\begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\begin{bmatrix} x^{k+1} \\ v \end{bmatrix}=\begin{bmatrix} P+\frac{1}{\lambda}I & A^T \\ A & 0 \end{bmatrix}^{-1}\begin{bmatrix} \frac{1}{\lambda}(z^k-u^k)-q \\ b \end{bmatrix}----------(1)$$

$$z^{k+1}=\left(x^{k+1}+u^k\right)_+ ----------(2)$$

$$u^{k+1}=u^k+x^{k+1}-z^{k+1} ----------(3)$$

Plot:

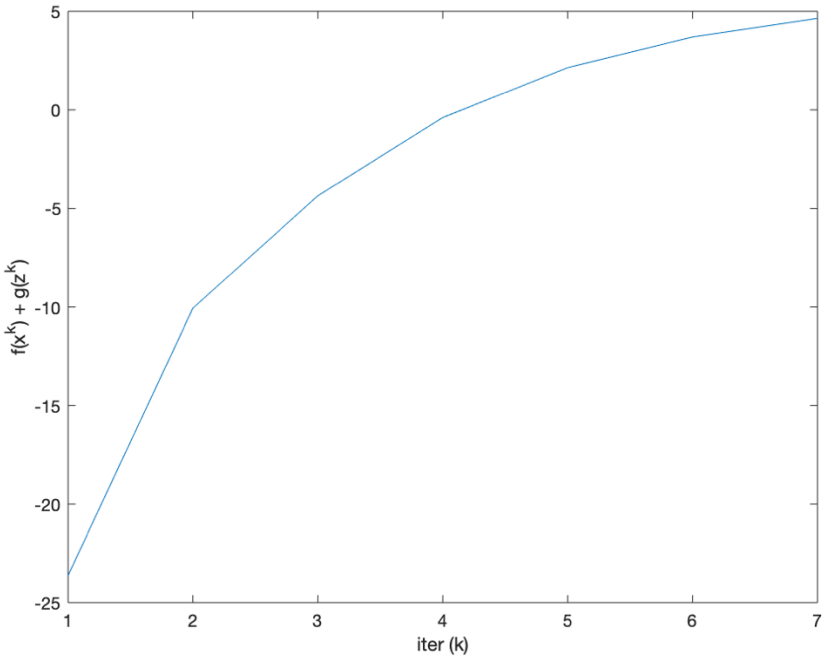


FIGURE 14

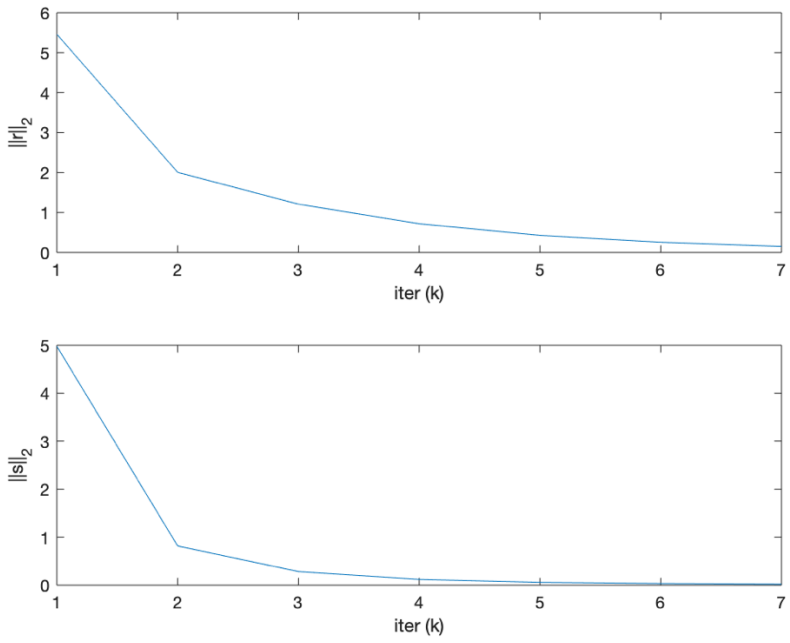


FIGURE 15

Code:

```
randn('state', 0);
rand('state', 0);

n = 100;

% generate a well-conditioned positive definite matrix
% (for faster convergence)
P = rand(n);
P = P + P';
[V D] = eig(P);
P = V*diag(1+rand(n,1))*V';

q = randn(n,1);
r = randn(1);

l = randn(n,1);
u = randn(n,1);
lb = min(l,u);
ub = max(l,u);
[x z u r_norm s_norm g obj] = quadprog(P, q, r, lb, ub, 1.0, 1.0);
fprintf('%20s\t%10s\t%10s\t \n', 'x^(k+1)','z^(k+1)','u^(k+1)');
for i =1:length(x)
fprintf('%20f\t%10.4f\t%10.4f\t \n', x(i),z(i),u(i));
end

figure
%plotting difference between current z and old z
subplot(2,1,1)
plot(1:g,r_norm)
ylabel('||r||_2'); xlabel('iter (k)');
%plotting difference between z and x
subplot(2,1,2)
plot(1:g,s_norm)
ylabel('||s||_2'); xlabel('iter (k)');
figure
%plotting norm with iteration
plot(1:g,obj)
ylabel('f(x^k) + g(z^k)'); xlabel('iter (k)');
function [x_hat z u r_norm s_norm g obj] = quadprog(P, q, r, lb, ub, rho,
alpha)
t_start = tic;
QUIET    = 0;
MAX_ITER = 1000;
ABSTOL   = 1e-4;
RELTOL   = 1e-2;
n = size(P,1);
x = zeros(n,1);
z = zeros(n,1);
u = zeros(n,1);

for k = 1:MAX_ITER

    if k > 1
        x = R \ (R' \ (rho*(z - u) - q));
    else
        R = chol(P + rho*eye(n));
        x = R \ (R' \ (rho*(z - u) - q));
    end

    % z-update with relaxation
```

```
    zold = z;
    x_hat = alpha*x +(1-alpha)*zold;
    z = min(ub, max(lb, x_hat + u));
    % u-update
    u = u + (x_hat - z);

    % diagnostics, reporting, termination checks
    obj(k)  = objective(P, q, r, x);

    r_norm(k)  = norm(x - z);
    s_norm(k)  = norm(-rho*(z - zold));
    if(z-zold<RELTOL )
         g=k
         break;
     end
end

if ~QUIET
    toc(t_start);
end
end

function obj = objective(P, q, r, x)
    obj = 0.5*x'*P*x + q'*x + r;
end
```